Certainly! Here's a simple C program to find the middle element of a singly linked list:

```c
#include <stdio.h>
#include <stdlib.h>

// Define a structure for a node in the linked list
struct Node {
    int data;
    struct Node* next;
};

// Function to find the middle element of the linked list
int findMiddle(struct Node* head) {
    struct Node* slow_ptr = head;
    struct Node* fast_ptr = head;

    if (head != NULL) {
        while (fast_ptr != NULL && fast_ptr->next != NULL) {
            fast_ptr = fast_ptr->next->next;
            slow_ptr = slow_ptr->next;
        }
        return slow_ptr->data;
    }
    return -1; // Return -1 if the list is empty
}

// Function to insert a new node at the end of the linked list
void insertAtEnd(struct Node** head, int data) {
    struct Node* new_node = (struct Node*)malloc(sizeof(struct Node));
    struct Node* last = *head;

    new_node->data = data;
    new_node->next = NULL;

    if (*head == NULL) {
        *head = new_node;
        return;
    }

    while (last->next != NULL) {
        last = last->next;
    }

    last->next = new_node;
}

// Function to display the linked list
```

```
void display(struct Node* head) {
    struct Node* current = head;

    while (current != NULL) {
        printf("%d -> ", current->data);
        current = current->next;
    }
    printf("NULL\n");
}

int main() {
    struct Node* head = NULL;

    // Insert elements into the linked list
    insertAtEnd(&head, 1);
    insertAtEnd(&head, 2);
    insertAtEnd(&head, 3);
    insertAtEnd(&head, 4);
    insertAtEnd(&head, 5);

    printf("Linked List: ");
    display(head);

    int middle = findMiddle(head);

    if (middle != -1) {
        printf("Middle element: %d\n", middle);
    } else {
        printf("The list is empty.\n");
    }

    return 0;
}
```

This program defines a `Node` structure for the linked list, provides functions to insert elements at the end and find the middle element, and demonstrates its usage in the `main` function.